ENVISIONING SUCCESS: PREDICTING PERMANENT MAGNET RESISTANCE OF ELECTRIC MOTOR USING MACHINE LEARNING

AN INDUSTRY ORIENTED MINI PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAB

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

BOLLABOINA RUCHITH	**21UK1A05H0**
SENU SAI KIRAN	**21UK1A05F2**
BEEMARAPU KAMAL	**21UK1A05H3**
YATA HARISH	**21UK1A05J2**

Under the guidance of

P.IILANNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTU, HYDERABAD

BOLLIKUNTA, WARANGAL (T.G) - 506005

2021 - 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE
BOLLIKUNTA, WARANGAL – 506005

2020 - 2024

<u>CERTIFICATE</u>

This is to certify that the Industry Oriented Mini Project entitled " ENVISIONING SUCCESS: PREDICTING PERMANENT MAGNET RESISTANCE OF ELECTRIC MOTOR USING MACHINE LEARNING" is being Submitted   **by**

BOLLABOINA RUCHITH (21UK1A05H0) , SENU SAI KIRAN (21UK1A05F2) ,

BEEMARAPU KAMAL (21UK1A05H3) , YATA HARISH (21UK1A05J2) ,

in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** during the academic year **2024 – 2025** , is a record of work carried out bythem under the guidance and supervision.

Project Guide   **Head of the Department**

P.IILANNA   **Dr.R. NAVEEN KUMAR**

                                                                                                          (Professor)

                                          EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this Industry Oriented Mini Project in the institute.

We extend our heartful thanks to **Dr.R. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the Industry Oriented Mini Project.

We express heartful thanks to Smart Educational Services Limited, for their constant supervision as well as for providing necessary information regarding the Industry Oriented Mini Project and for their support in completing the Industry Oriented Mini Project.

We express heartful thanks to the guide**, P.IILANNA** , Head of the Department of CSE for his constant support and giving necessary guidance for completion of this Industry Oriented Mini Project.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

BOLLABOINA RUCHITHA   21UK1A05H0

SENU SAI KIRAN   21UK1A05F2

BEEMARAPU KAMAL   21UK1A05H3

YATA HARISH   21UK1A05J2

ABSTRACT

This study explores the application of machine learning techniques for predicting the permanent magnet resistance of electric motors. Permanent magnet resistance is a critical parameter affecting the efficiency and performance of electric motors, especially in various industrial and automotive applications. Traditional methods for determining this resistance involve complex physical modeling and experimental testing,which can be time-consuming and costly.

In this research, machine learning models are trained using datasets containing relevant motor characteristics and corresponding resistance values. Various regression algorithms such as linear regression, decision trees, and neural networks are employed to develop predictive models.Feature selection techniques are applied to identify the most influential motor parameters affecting resistance. The performance of these models is evaluated using metrics like mean absolute error,mean squared error, and R-squared.

The results demonstrate the effectiveness of machine learning in accurately predicting permanent magnet resistance based on motor specifications. The developed models offer a promising alternative to traditional methods by providing rapid and cost-effective predictions. This approach can significantly contribute to optimizing motor design and enhancing overall efficiency in electric vehicles, industrial automation, and renewable energy systems.

Keywords: machine learning, electric motors, permanent magnet resistance, predictive modeling

# CONTENTS

1. INTRODUCTION

1.1 Project Overview

The permanent-magnet synchronous machine (PMSM) drive is one of the best choices for a full range of motion control applications. For example, the PMSM is widely used in robotics, machine tools, actuators, and it is being considered in high-power applications such as industrial drives and vehicular propulsion. It is also used for residential/commercial applications. The PMSM is known for having low torque ripple, superior dynamic performance, high efficiency, and high power density.

The task is to design a model with appropriate feature engineering that estimates the target temperature of a rotor.In this project, we will be using algorithms such as Linear Regression, Decision Tree, Random Forest and SVM. We will train and test the data with these algorithms and select the best model. The best algorithm will be selected and saved in pkl format. We will be doing flask integration and IBM deployment.
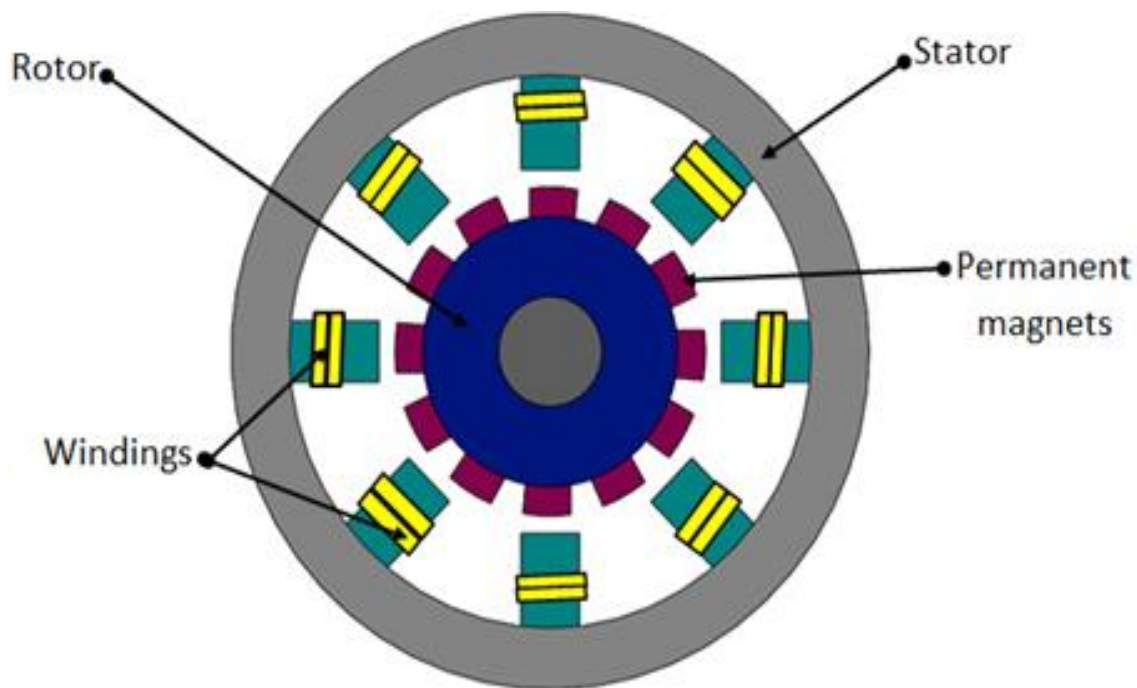
Technical Architecture:
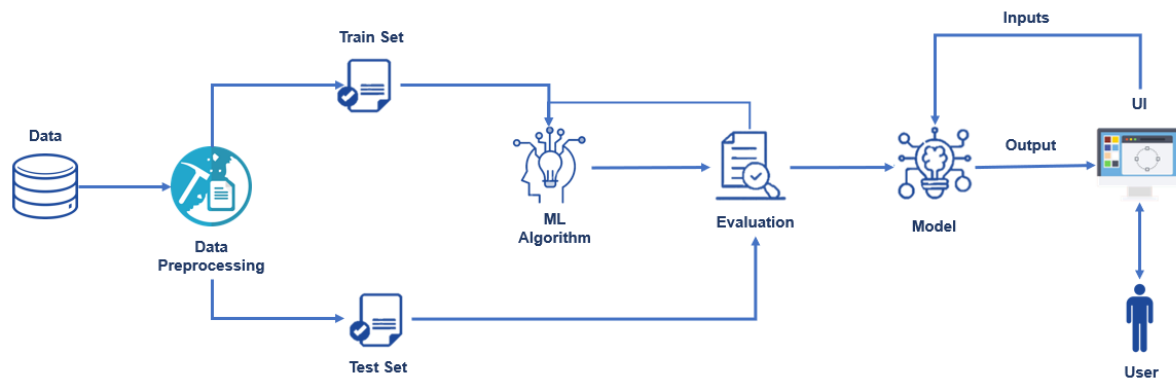
1.2 :

The project's main purpose is to assist prospective students in making informed decisions about their education by predicting university scores based on various characteristics. This can help students choose universities that align with their goals and expectations. Additionally, it can provide valuable feedback to universities and impact their funding, reputation, and overall quality.

2. LITERATURE SURVEY

A literature survey for a university score prediction project would involve researching and reviewing existing studies, articles, and other publications on the topic of university rankings. The survey would aim to gather information on current prediction systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous score prediction projects, and any relevant data or findings that could inform the design and implementation of the current project.

2.1 Existing Problem (or) problem statement

Predicting university scores with machine learning is a challenging problem that holds significant importance for both students and educational institutions. Envisioning success in this context means developing accurate models that can forecast a student's academic performance based on various input factors, such as high school grades, standardized test scores, and extracurricular activities. The difficulty lies in accounting for the multifaceted nature of academic achievement, which is influenced by numerous variables, including personal motivation and learning styles. Moreover, the data used for prediction may be incomplete or biased, making it essential to create robust models that can handle these complexities.

2.2 Proposed solution:

The goal of this project is to develop a web-based system that predicts university scores based on various characteristics. This system will serve as a tool for prospective students to make informed decisions about their higher education choices while also providing valuable feedback to universities for improvement.

1.Understanding the Problem Domain

2. Key Concepts and Terminology

3. Literature Search Strategy

4. Review of Existing Research

5. Analysis of Methodologies

6. Gaps and Opportunities

1.Understanding the Problem Domain

Electric Motors and Magnet Resistance: Familiarize yourself with the components of electric motors, particularly permanent magnets and their role in motor efficiency and performance.
Machine Learning in Predictive Maintenance: Review how machine learning is applied in predicting faults, performance degradation, and efficiency in industrial equipment, including electric motors.

2. Key Concepts and Terminology

Magnet Resistance: Define magnet resistance in the context of electric motors. Understand its significance in more efficiency and operational longevity.
Feature Engineering: Explore relevant features such as temperature, current, voltage, and mechanical stress that the impact magnet resistance.

3. Literature Search Strategy

Databases and Sources: Utilize academic databases (IEEE Xplore, ScienceDirect, ACM Digital Library), the conference proceedings, and relevant journals (IEEE Transactions on Industrial Electronics, Mechanical Systems and Signal Processing) for recent research.
Search Keywords: Use combinations like "electric motor," "permanent magnet," "magnet resistance," "predictive maintenance," and "machine learning."

4. Review of Existing Research

Studies on Motor Efficiency: Identify studies that focus on improving efficiency through predictive the techniques related to magnet resistance.
Machine Learning Techniques: Look for papers that apply regression, classification, anomaly detection, or time-series analysis to predict magnet resistance or related parameters.
Case Studies and Applications: Find practical applications or case studies where machine learning the models have been successfully deployed for predicting motor performance.

5. Analysis of Methodologies

Machine Learning Models: Compare different models (e.g., linear regression, neural networks, support vector machines) applied to similar predictive maintenance tasks.
Data Preprocessing: Examine how researchers preprocess data (feature scaling, normalization) to the enhance model accuracy.
Performance Metrics: Evaluate metrics used to assess model performance (e.g., RMSE, MAE, R-squared) in predicting magnet resistance.

6. Gaps and Opportunities

Identify Research Gaps: Determine areas where existing literature may be lacking, such as specific types of motors, environmental conditions, or integration of real-time sensor data.
Future Directions: Suggest potential avenues for future research, such as hybrid models combining physics.
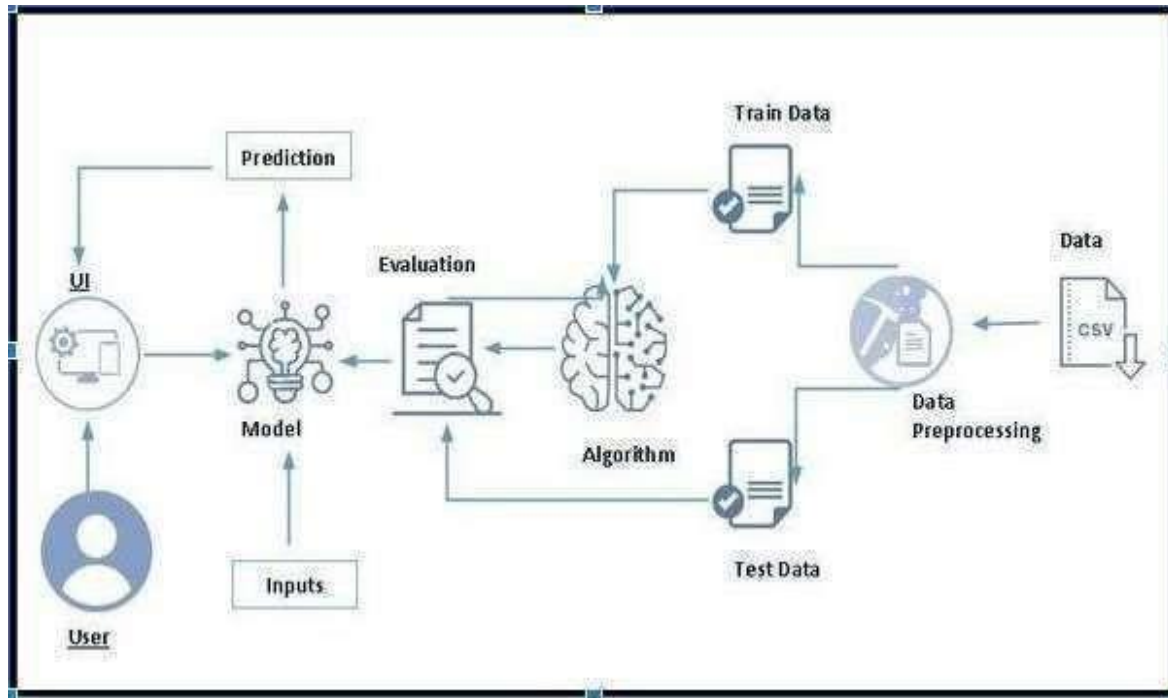
3. THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM

Figure 1: Technical Architecture

3.2 Hardware /Software designing

Hardware design

1.Server/Cloud Infrastructure: Host the web application and machine learning model on a server or cloudbased infrastructure. Cloud platforms like AWS, Azure, or GCP are common choices.

2.Compute Resources: Ensure sufficient CPU and memory resources to handle user interactions and model predictions, with scalability options to accommodate increased traffic.

3.Storage: Store the HTML templates, Python scripts, and the saved machine learning model (usp.pkl) on the server or cloud storage. Consider scalable storage options for datasets.

4.Network Infrastructure: Ensure a reliable network connection to handle user requests, data transfer between the web application and the machine learning model, and database connectivity if applicable.

5.Security Measures: Implement security measures to protect user data, ensure the confidentiality

and integrity of the application and data, and secure communication between components.

3.3Software design

1.Web Application (Frontend): Use HTML, CSS, and JavaScript to design the user interface for the web application. The Flask web framework will serve these templates.

2.Web Application (Backend): Implement the backend using Flask, a Python web framework. It handles user interactions, receives input, and sends requests to the machine learning model for predictions.

3.Machine Learning Model: The machine learning model is responsible for predicting university scores. Use Python for model development with libraries like Scikit-Learn, Numpy, and Pandas.

4.Database (Optional): If data storage is necessary, consider using a relational database system like MySQL or PostgreSQL for managing and retrieving data.

5.Version Control: Use version control systems like Git to track changes in the codebase and collaborate with team members, if applicable.

6.Testing and Quality Assurance: Implement testing methodologies to ensure the correctness and reliability of the web application and machine learning model.

7.Model Deployment: Deploy the machine learning model using Flask. The usp.pkl file can be loaded within the Flask application to make predictions.

8.Security Measures: Implement security practices to protect against common web application vulnerabilities, including input validation, and secure sensitive data.

9.Documentation: Create documentation that outlines the project's architecture, installation instructions, and usage guidelines.

10.Monitoring and Logging: Implement monitoring and logging solutions to track the application's performances, diagnose issues, and identify opportunities for optimization.

11.Scaling Strategy: Develop a strategy for scaling the application in case of increased user demand, which may involve load balancing and redundancy.

12.Deployment and Maintenance: Plan for deployment, regular updates, and maintenance to ensure the application remains secure and operational.

Overall, the hardware and software design for this project is essential to ensure a reliable, efficient, and secure system for predicting university scores with machine learning. The choice of specific technologies and configurations will depend on the project's scale, budget, and technical requirements.

PROJECT FLOW

1. Start

2. Data Collection:

   - Collect historical data on electric motor operation.

   - Include variables like temperature, current, voltage, speed, and magnet resistance.

3. Data Preprocessing:

   - Clean the data (remove outliers, handle missing values).

   - Normalize or scale the data if necessary.

4. Feature Engineering:

   - Identify relevant features impacting magnet resistance.

   - Create new features if needed (e.g., derived features from existing data).

5. Split Data:

   - Divide the dataset into training and testing sets.

6. Model Selection

 - Choose appropriate machine learning algorithms

   (e.g., linear regression, decision trees, neural networks).

7. Model Training:

   - Train the selected model using the training dataset.

8. Model Evaluation:

   - Evaluate model performance using metrics such as Mean Squared Error (MSE),

     Root Mean Squared Error (RMSE), or R-squared.

   - Optimize hyperparameters if necessary.

9. Predict Magnet Resistance:

   - Use the trained model to predict magnet resistance for new data instances.

10. Monitor Performance:

   - Monitor predictions against actual magnet resistance measurements.

   - Ensure model accuracy and reliability.

11. Deployment:

   - Deploy the model for real-time predictions or periodic updates.

   - Integrate into operational processes for predictive maintenance.

12. End

Flowchart Elements:

- **Start/End Symbol:** Oval shapes with "Start" and "End" labels.
- **Process Symbols:** Rectangles for steps like Data Collection, Data Preprocessing,
- Feature Engineering, etc.
- **Decision Symbols:** Diamonds for decision points (e.g., Model Selection).
- **Arrows:** Connectors between symbols to indicate the flow of steps.
- **Image Inserts:** Use icons or images of data, graphs, machine learning algorithms
- (like neural network icons), and performance metrics (like graphs showing evaluation results).

4. DATA COLLECTION AND PREPARATION

Data Collection: Collect the datasets from different sources and import necessary

Visualizing And Analyzing The Data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.
Note: There is a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing The Libraries

Import the necessary libraries as shown in the image To know about the packages refer to the link given on prerequisites.

```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

-

.

Read The Datasets

The dataset is read as a data frame (df in our application) using the pandas' library (pd is the alias name given to the pandas package).

```
In [4]: df = pd.read_csv('pmsm_temperature_data.csv')
        df.head()
```

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_
0	-0.752143	-1.118446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.0180
1	-0.771263	-1.117021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.0176
2	-0.782892	-1.116681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.0173
3	-0.780935	-1.116764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.0176
4	-0.774043	-1.116775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.0181

Descriptive Analysis

We'll see which particular variables contribute to the rotor temperature individually by checking their statistical significance.

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max, and percentile values of continuous features.

df.info():

This function is used to display a brief introduction about the data set such as the. of rows and columns, the Data type of each column, whether the null values are present in the column or not.

```
In [8]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        Int64Index: 942677 entries, 0 to 982769
        Data columns (total 13 columns):
        ambient          942677 non-null float64
        coolant          942677 non-null float64
        u_d              942677 non-null float64
        u_q              942677 non-null float64
        motor_speed      942677 non-null float64
        torque           942677 non-null float64
        i_d              942677 non-null float64
        i_q              942677 non-null float64
        pm               942677 non-null float64
        stator_yoke      942677 non-null float64
        stator_tooth     942677 non-null float64
        stator_winding   942677 non-null float64
        profile_id       942677 non-null int64
        dtypes: float64(12), int64(1)
        memory usage: 100.7 MB
```

df.describe()

This function is used to analyze the descriptive statistics of the data such as mean, median, quartile values, maximum and minimum values of each column.

```
In [9]: df.describe()
```

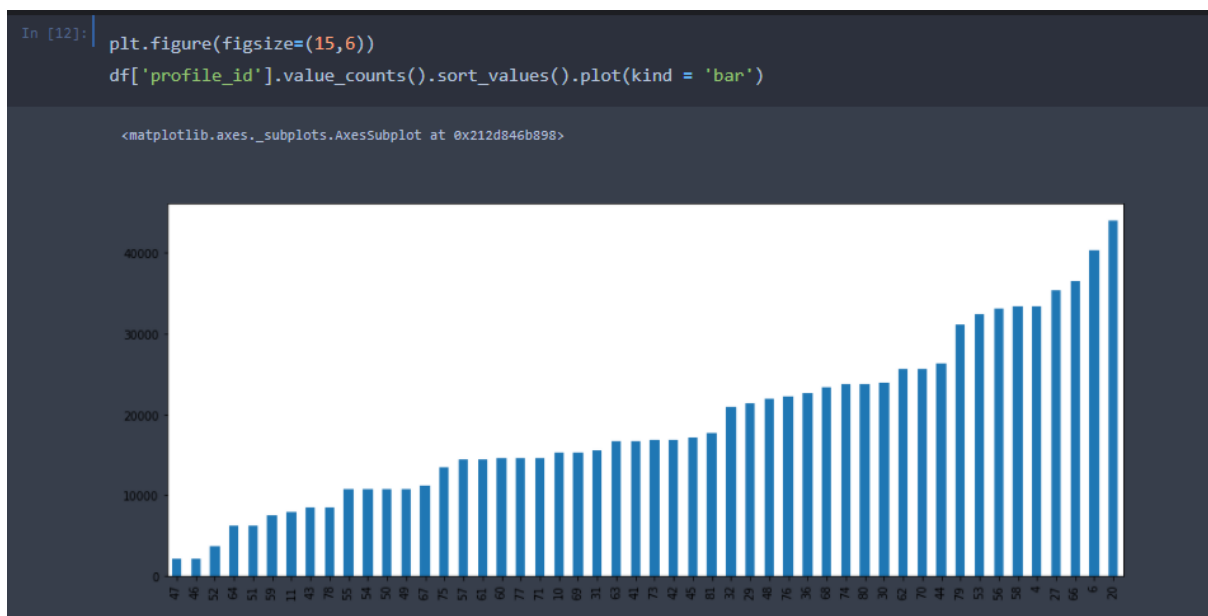|	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm
count	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.000000	942677.
mean	-0.030010	-0.008238	-0.021835	0.007720	0.003210	0.020170	-0.002465	0.019958	-0.00509
std	1.007729	1.009503	0.994308	0.996054	0.996456	0.999063	1.000106	0.999683	1.00141
min	-8.573954	-1.429349	-1.655373	-1.861463	-1.371529	-3.345953	-3.245874	-3.341639	-2.63199
25%	-0.634542	-1.041886	-0.854390	-0.881885	-0.951878	-0.265042	-0.760764	-0.254672	-0.66783
50%	0.242495	-0.185147	0.225416	-0.089520	-0.140245	-0.121022	0.196713	-0.091449	0.09915
75%	0.681905	0.698607	0.356361	0.859836	0.855827	0.561778	1.013952	0.543750	0.67784
max	2.967117	2.649032	2.274734	1.793498	2.024164	3.016971	1.060937	2.914185	2.91745

5.EXPLORATORY DATA ANALYSIS

Uni-Variate Analysis

Here we get to know about our data
Bar Graph:
A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.


```
In [12]:  plt.figure(figsize=(15,6))
          df['profile_id'].value_counts().sort_values().plot(kind = 'bar')

          <matplotlib.axes._subplots.AxesSubplot at 0x212d846b898>
```

.

As we can see, session ids 66, 6, and 20 have the most number of measurements recorded
·Box plot:
A boxplot is a standardized way of displaying the distribution of data based on a five-number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can tell you about your outliers and what their values are.

Distribution plot:
The distribution plot is suitable for comparing range and distribution for groups of numerical data.
Data is plotted as value points along an axis.

```
In [13]:  df.columns

          Index(['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'torque', 'i_d',
                 'i_q', 'pm', 'stator_yoke', 'stator_tooth', 'stator_winding',
                 'profile_id'],
                dtype='object')

In [14]:  #Plotting Distribution and Boxplot for all the features to check for skewness

In [15]:  for i in df.columns:
              sns.distplot(df[i],color='g')
              sns.boxplot(df[i],color = 'y')
              plt.vlines(df[i].mean(),ymin = -1,ymax = 1,color = 'r')#drawing the mean line
              plt.show()
```
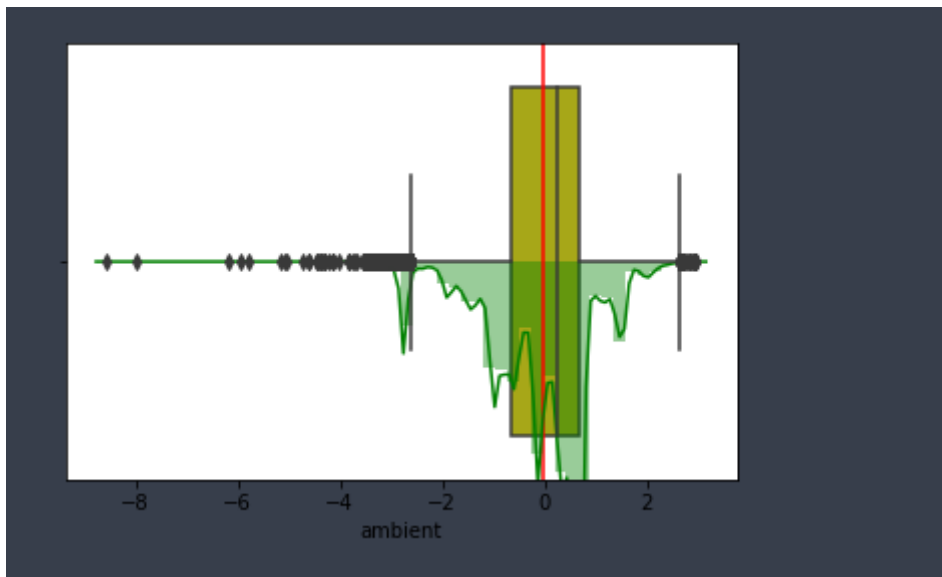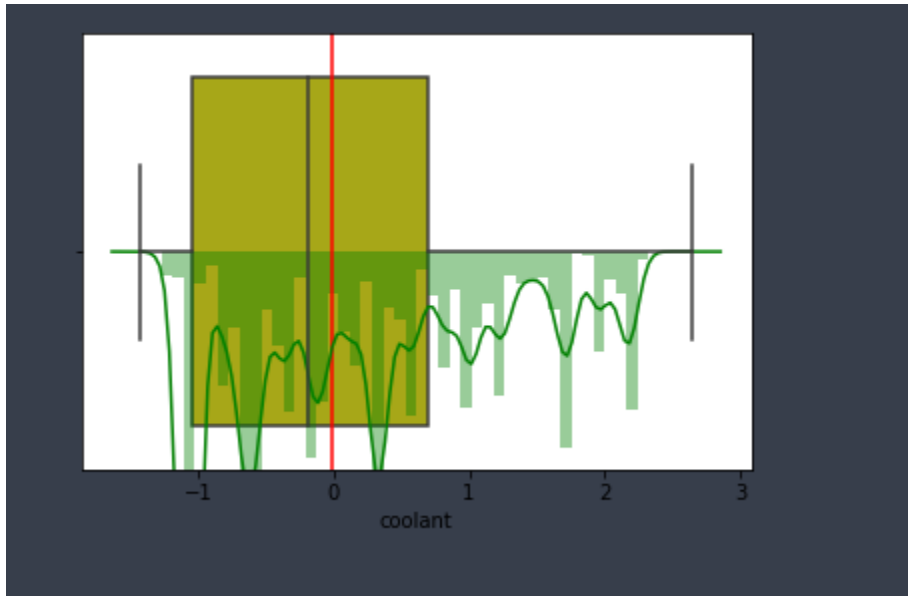
All features boxplots are plotted and the following conclusions are drawn.
As we can see from the above plots, the mean and median for most of the plots are very close to each other. So the data seems to have low skewness for almost all variables .

Multi-Variate Analysis

Multivariate analysis (MVA) is a Statistical procedure for the analysis of data involving more than one type of measurement or observation. It may also mean solving problems where more than one dependent variable is analyzed simultaneously with other variables.

Scatterplot:
A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

As we want to predict the temperatures of stator components and rotor(pm), we will drop these values from our dataset for regression. Also, torque is a quantity, which is not reliably measurable in field applications, so this feature shall be omitted in this modeling.

```
fig, axes = plt.subplots(2, 4, figsize=(20, 5),sharey=True)
sns.scatterplot(df['ambient'],df['pm'],ax=axes[0][0])
sns.scatterplot(df['coolant'],df['pm'],ax=axes[0][1])
sns.scatterplot(df['motor_speed'],df['pm'],ax=axes[0][2])
sns.scatterplot(df['i_d'],df['pm'],ax=axes[0][3])
sns.scatterplot(df['u_q'],df['pm'],ax=axes[1][0])
sns.scatterplot(df['u_d'],df['pm'],ax=axes[1][1])
sns.scatterplot(df['i_q'],df['pm'],ax=axes[1][2])
```

<matplotlib.axes._subplots.AxesSubplot at 0x212d84f4f60>

Heat-map:

A heat map is a data visualization technique that shows the magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.

```
plt.figure(figsize=(14,7))
sns.heatmap(df.corr(),annot=True);
```
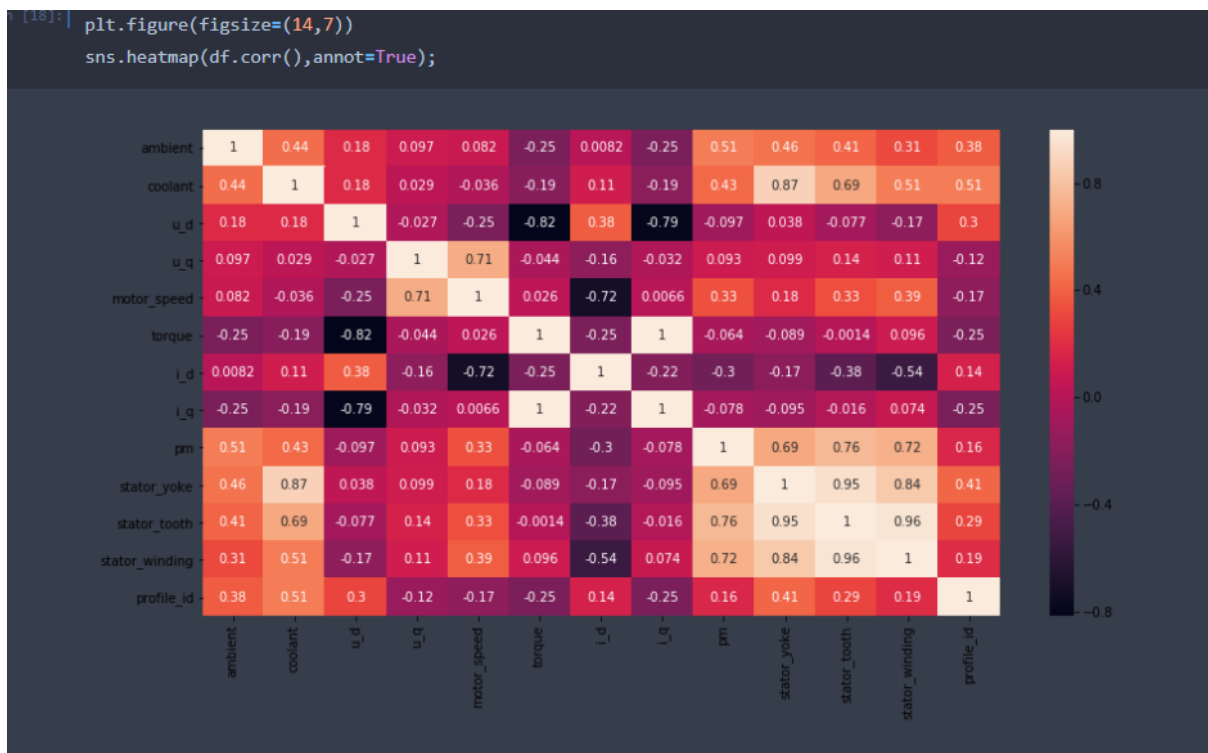
From the heatmap above, we can see that torque and q component of current are almost perfectly correlated. Also, there seems to be a very high correlation between temperature measurements of stator yoke, stator tooth, and stator windings.
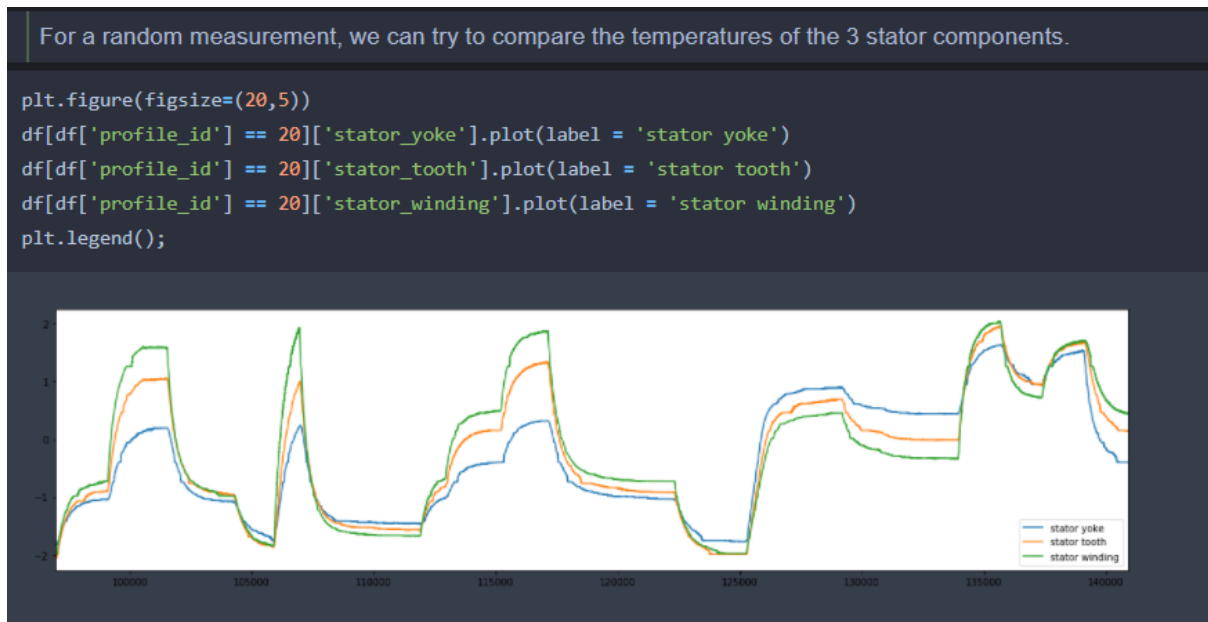
As we can see from the plot, all three stator components follow a similar measurement variance. As the dataset author mentioned, the records in the same profile id have been sorted by time, we can assume that these recordings have been arranged a series of times.

Due to this, we can infer that there has not been much time given for the motor to cool down in between recording the sensor data as we can see that initially the stator yoke temperature is low as compared to the temperature of stator winding but as we progress in time, the stator yoke temperature goes above the temperature of the stator winding.

As profile_id is an id for each measurement session, we can remove it from any further analysis and model building.

```python
In [20]:  df.drop('profile_id',axis = 1,inplace=True)
          df_test.drop('profile_id',axis = 1,inplace=True)
```

Data Pre-Processing:

As we have understood how the data is. Let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps:

Handling missing values
Handling categorical data
Handling outliers
Scaling Techniques
Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

In the data frame, head() function is used to display the first 5 data. Our dataset has ambient, coolant,u_d,u_q,motor_speed , i_d , i_q ,stator_yoke,stator_winding and profile_id, pm(output)

```
df.head()
```

ant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth	stator_winding	profile_id
446	0.327935	-1.297858	-1.222428	-0.250182	1.029572	-0.245860	-2.522071	-1.831422	-2.066143	-2.018033	4
021	0.329665	-1.297686	-1.222429	-0.249133	1.029509	-0.245832	-2.522418	-1.830969	-2.064859	-2.017631	4
681	0.332771	-1.301822	-1.222428	-0.249431	1.029448	-0.245818	-2.522673	-1.830400	-2.064073	-2.017343	4
764	0.333700	-1.301852	-1.222430	-0.248636	1.032845	-0.246955	-2.521639	-1.830333	-2.063137	-2.017632	4
775	0.335206	-1.303118	-1.222429	-0.248701	1.031807	-0.246610	-2.521900	-1.830498	-2.062795	-2.018145	4

Drop Unwanted Features

As we want to predict the temperatures of stator components and rotor(pm), we will drop these values from our dataset for regression. Also, torque is a quantity, which is not reliably measurable in field applications, so this feature shall be omitted in this modeling.

Dropping the columns from the dataset is being concluded with the help of a scatter plot, which is available in the data analysis part.

```
df.drop(['stator_yoke','stator_tooth','stator_winding','torque'],axis = 1)
```

Handling Missing Values

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that the education column and previous year rating column have null values.

```
In [11]: df.isnull().sum()

         ambient          0
         coolant          0
         u_d              0
         u_q              0
         motor_speed      0
         torque           0
         i_d              0
         i_q              0
         pm               0
         stator_yoke      0
         stator_tooth     0
         stator_winding   0
         profile_id       0
         dtype: int64
```

There are no null values in the dataset, we can skip this step.

6.MODEL BUILDING

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying four Regression algorithms. The best model is saved based on its performance. To evaluate the performance of the model, we use root mean square error and r-square value

Linear Regression model

A function named LinearRegression is created and train and test data are passed as the parameters. Inside the function, the LinearRegression algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the performance of the model, we use root mean square error and r-square value

```
In [51]:  from sklearn.linear_model import LinearRegression
          from sklearn.tree import DecisionTreeRegressor
          from sklearn.ensemble import RandomForestRegressor
          from sklearn.svm import SVR

In [52]:  lr=LinearRegression()
          dr=DecisionTreeRegressor()
          rf =RandomForestRegressor()
          svm =SVR()
```

```
In [*]:  lr.fit(X_train,y_train)
         dr.fit(X_train,y_train)
         rf.fit(X_train,y_train)
         svm.fit(X_train,y_train)
```

Decision Tree Model

A function named decision tree is created and train and test data are passed as the parameters. Inside the function, the DecisionTreeRegressor algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the performance of the model, we use root mean square error and r-square value.

```
In [51]:  from sklearn.linear_model import LinearRegression
          from sklearn.tree import DecisionTreeRegressor
          from sklearn.ensemble import RandomForestRegressor
          from sklearn.svm import SVR
```

```
In [52]:  lr=LinearRegression()
          dr=DecisionTreeRegressor()
          rf =RandomForestRegressor()
          svm =SVR()
```

```
In [*]:  lr.fit(X_train,y_train)
         dr.fit(X_train,y_train)
         rf.fit(X_train,y_train)
         svm.fit(X_train,y_train)
```

Random Forest Model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestRegressor algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluate the performance of the model, we use root mean square error and r-square value.

7. TESTING MODEL WITH MULTIPLE EVALUATION METRICS

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for regression tasks including Mean absolute Error, Mean Squared Error, Root Mean Squared Error and R square Score.

Compare The Model

```
from sklearn import metrics

print(metrics.r2_score(y_test,p1))
print(metrics.r2_score(y_test,p2))
print(metrics.r2_score(y_test,p3))
print(metrics.r2_score(y_test,p4))


 0.9698725757690718
 0.47757469778170836
```

Out of all the models. The decision Tree regressor is giving an r2-score of 96%, it means the model is able to explain 96% of the data. so we will select the decision tree model and save it.
 fit() function. Test data is predicted with .predict() function and saved in new variable .

with random forest algorithm. You can test with all algorithm .With the help ofpredict() function.

After calling the function the results of models are displayed as output . From the three models random forest isperforming well.

Evaluating Performance Of The Model

Evaluating the model by using RMSE (Root Mean Squared Error)

```
In [20]:  from sklearn.metrics import mean_squared_error

In [26]:  print(mean_squared_error(y_test,p1))

          0.030187256377134934
```

From the above picture, we can infer that the RMSE value is 0.03, which is very low.It means our predicted values and actual values are almost equal. The difference between actual values and predicted value

Normalizing The Values

As we want to predict the temperatures of stator components and rotor(pm), we will drop these values from our dataset for regression. Also, torque is a quantity, which is not reliably measurable in field applications, so this feature shall be omitted in this modeling.
We are using min max scaler ,which is a function in preprocessing module in sklearn library is very less. so we are considering this model.

```
In [52]:  mm = MinMaxScaler()
          X = mm.fit_transform(X)
          X_df_test = mm.fit_transform(X_df_test)
          y = df['pm']
          y_df_test = df_test['pm']
          X = pd.DataFrame(X,columns = ['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'i_d','i_q'])
          X_df_test = pd.DataFrame(X_df_test,columns = ['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'i_d','i_q
          y.reset_index(drop = True,inplace = True)
          y_df_test.reset_index(drop = True,inplace = True)
```

Splitting Data Into Train And Test

Now let's split the Dataset into train and test sets. For splitting training and testing data, we are using the train_test_split() function from sklearn. As parameters, we are passing x_resample, y_resample, test_size, random_state.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

8.MODEL DEPLOYMENT

Save The Best Model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```python
filename = 'model.pkl'
pickle.dump(rf,open(filename,'wb'))
```

Save the model

```python
In [27]: import joblib

In [65]: joblib.dump(dr,"model.save")

         ['model.save']
```

9.INTEGRATE WITH FRAMEWORK

Application Building

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script

After the model is built, we will be integrating it into a web application

Build The Python Flask App

In the flask application, the user values are taken from the HTML page

```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib
app = Flask(__name__)
model = joblib.load("model.save")
trans=joblib.load('transform.save')


app = Flask(__name__)
```

Load the home page

```python
app = Flask(__name__)

@app.route('/')
def predict():
    return render_template('Manual_predict.html')
```

Prediction function

```
@app.route('/y_predict',methods=['POST'])
def y_predict():
    x_test = [[float(x) for x in request.form.values()]]
    print('actual',x_test)
    x_test=trans.transform(x_test)
    print(x_test)
    pred = model.predict(x_test)

    return render_template('Manual_predict.html', prediction_text=('Permanent Magnet surface temperature: ',pred[0]))


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Build An HTML Page

We Build an HTML page to take the values from the user in a form and upon clicking on the predict button we get the temperature predicted. The values predicted are normalized values according to the dataset. Hence units are not considered. You can get these files from the project folder.

 Building Html Pages:

For this project, create five HTML files namely and save them in the templates folder.

For more information regarding HTML homepage: C:\Users\Dell\OneDrive\Desktop\project\Flask\templates

For more information regarding HTML index_page: C:\Users\Dell\OneDrive\Desktop\project\Flask\templates

For more information regarding HTML Manual_home: C:\Users\Dell\OneDrive\Desktop\project\Flask\templates

For more information regarding HTML home_predict: C:\Users\Dell\OneDrive\Desktop\project\Flask\templates

For more information regarding HTML result page: C:\Users\Dell\OneDrive\Desktop\project\Flask\templates

Let's see how our home.html page looks like

```
(base) PS C:\Users\Dell> cd C:\Users\Dell\OneDrive\Desktop\project\Flask
(base) PS C:\Users\Dell\OneDrive\Desktop\project\Flask> conda activate myenv
(myenv) PS C:\Users\Dell\OneDrive\Desktop\project\Flask> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 549-162-737
```

```python
from flask import Flask, render_template, request

app = Flask(__name__)

# Load the model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

# Define the route for the prediction form
@app.route('/predict', methods=['POST'])
def predict():
    # Get input values from the form
    u_q = float(request.form['u_q'])
    coolant = float(request.form['coolant'])
    u_d = float(request.form['u_d'])
    motor_speed = float(request.form['motor_speed'])
    i_d = float(request.form['i_d'])
    i_q = float(request.form['i_q'])
    ambient = float(request.form['ambient'])
    profile_id = float(request.form['profile_id'])
    stator_winding = float(request.form['stator_winding'])
    stator_tooth = float(request.form['stator_tooth'])
    stator_yoke = float(request.form['stator_yoke'])
    torque = float(request.form['torque'])

    # Make a prediction using the loaded model
    prediction = model.predict([[u_q, coolant, stator_winding, u_d, stator_tooth,
                                 motor_speed, i_d, i_q, stator_yoke, ambient, torque,
profile_id]])

    # Render the result template with the prediction
    return render_template('result.html', prediction=prediction)

# Define other routes as needed (e.g., home page)
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/home')
def home():
    return render_template('home.html')

if __name__ == '__main__':
    app.run(debug=True)
```
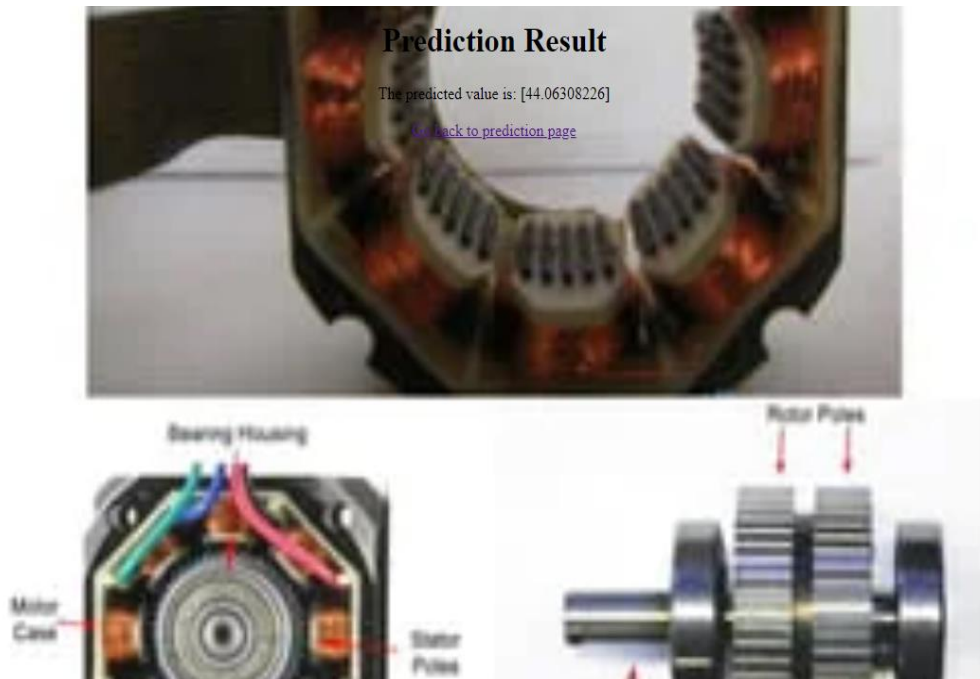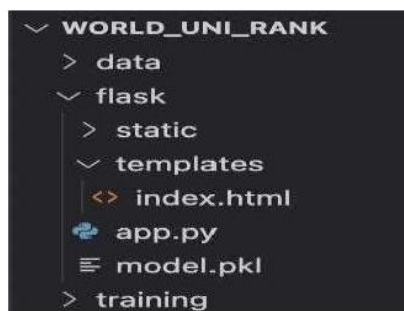
10.RESULT

Now, Go the web browser and write the localhost url(http://127.0.0.1:5000) to get the below result

Project Structure:

Create the Project folder which contains files as shown   below

We are building a flask application which needs HTML pages stored in the templatesfolder and a python script app.py for scripting.

- usp.pkl is our saved model. Further we will use this model for flask integration.

- Training folder contains a model training file.

11.ADVANTAGES AND DISADVANTAGES

Advantages

1.**Improved Accuracy**: ML models can potentially provide more accurate predictions of permanent magnet resistance compared to traditional analytical methods. They can learn from data patterns and relationships that might be difficult to capture through.

2.**Handling Complex Relationships**: Electric motors have intricate dynamics and interactions between components. ML models, especially advanced techniques like neural networks, can handle these nonlinear relationships effectively.

3. **Data Utilization**: ML thrives on data. By utilizing large datasets containing diverse motor   characteristics, operational conditions, and environmental factors, ML models can make predictions based on a broader range of inputs.

4.**Automation and Efficiency**: Once trained, ML models can automate the prediction process, reducing the need for manual calculations and potentially speeding up motor design and optimization processes

5.**Adaptability to New Data**: ML models can be continuously updated with new data,   allowing them to adapt to changes in motor design, manufacturing techniques, or operational conditions. This adaptability can lead to improved prediction accuracy over time

6.**Insights and Decision Support**: ML models can provide insights into the factors influencing permanent magnet resistance, aiding engineers in optimizing motor performance and identifying potential issues early on.

Disadvantages:

 1.**Data Quality and Availability**: ML models require large volumes of high-quality data for training. Obtaining such data can be challenging and may involve significant effort in data collection, cleaning, and preprocessing

2. **Model Complexity**: Some ML techniques, particularly deep learning models, can be complex and computationally intensive. This complexity may require substantial computational resources and expertise to develop, train, and deploy effectively.

 3. **Interpretability**: ML models, especially black-box models like deep neural networks, may lack interpretability. Understanding how the model arrives at its predictions can be difficult, which is crucial for industries requiring transparency and explainability in decision-making

4.**Overfitting**: Without proper regularization and validation, ML models can overfit to the training data, meaning they may perform well on training data but generalize poorly to new, unseen data. This is a significant concern in applications where accurate prediction across diverse motor types and conditions is essential

5.**Expertise Requirement**: Implementing ML models for predicting motor resistance requires expertise not only in machine learning but also in the domain of electric motors. Integrating ML into existing workflows and interpreting model outputs correctly requires specialized knowledge

6.**Maintenance and Updates**: ML models need ongoing maintenance to remain effective as operational conditions or motor designs change. This includes updating models with new data, retraining them periodically, and ensuring they continue to meet performance expectations.

12.APPLICATIONS

Design and Optimization:

1. **Performance Prediction**: ML models can predict the permanent magnet resistance during the design phase, aiding engineers in optimizing motor performance characteristics such as efficiency, torque output, and temperature management.
2. **Material Selection**: ML can assist in selecting the most suitable materials for permanent magnets based on their resistance properties under different operating conditions, optimizing the overall motor design.
3. **Parameter Sensitivity Analysis**: ML models can analyze how changes in motor parameters (such as geometry, windings, or cooling methods) affect permanent magnet resistance, helping designers make informed decisions.

Manufacturing and Quality Control:

1. **Fault Detection**: ML models can monitor and detect deviations in permanent magnet resistance during the manufacturing process. This can help identify potential defects or variations in production that could affect motor performance.
2. **Process Optimization**: ML can optimize manufacturing processes to ensure consistent and predictable permanent magnet resistance across batches of electric motors, improving overall quality control.

Operational Monitoring and Maintenance:

1. **Condition Monitoring**: ML models can continuously monitor the permanent magnet resistance during motor operation. Deviations from expected resistance values can indicate potential issues such as magnet degradation or overheating.
2. **Predictive Maintenance**: By predicting changes in permanent magnet resistance over time, ML can enable predictive maintenance strategies. Early identification of issues can prevent costly failures and extend motor lifespan.

Energy Efficiency and Performance Enhancement:

1. **Efficiency Optimization**: ML models can identify factors influencing permanent magnet resistance that impact motor efficiency. Insights gained can be used to optimize operating conditions or implement energy-saving strategies.
2. **Dynamic Performance Adjustment**: ML predictions of permanent magnet resistance can inform real-time adjustments in motor control strategies, enhancing dynamic performance and responsiveness.

Research and Development:

1. **New Motor Designs**: ML can accelerate the development of new motor designs by quickly evaluating the impact of design changes on permanent magnet resistance and overall performance metrics.
2. **Advanced Modeling**: ML techniques can be integrated into sophisticated models that simulate motor behavior under various conditions, enhancing understanding and enabling virtual prototyping.

Integration with IoT and Smart Systems:

1. **IoT Integration**: ML models predicting permanent magnet resistance can be integrated with IoT devices for real-time monitoring and control of electric motors in smart industrial systems.
2. **Data Analytics**: ML-driven analytics can leverage large volumes of operational data to continuously improve predictions and optimize motor performance in IoT-enabled environments.

.

13. CONCLUSION AND FUTURE WORK

Predicting permanent magnet resistance of electric motors using machine learning represents a significant advancement in motor design, manufacturing, and maintenance. By leveraging ML algorithms, engineers can achieve more accurate predictions of resistance values under varying operational conditions, leading to improved motor efficiency, reliability, and performance. Key benefits include:

- **Improved Accuracy**: ML models can capture complex relationships and patterns in data that traditional analytical methods may overlook, thereby enhancing prediction accuracy.
- **Efficiency Gains**: Automation of resistance prediction through ML reduces manual effort and accelerates decision-making in motor design and optimization processes.
- **Insights Generation**: ML provides insights into factors influencing permanent magnet resistance, aiding in better understanding motor behavior and enabling proactive maintenance strategies.
- **Operational Optimization**: By predicting resistance changes over time, ML supports predictive maintenance strategies that reduce downtime and extend motor lifespan.

Future Work:

1. **Enhanced Model Interpretability**: Addressing the interpretability of ML models remains crucial. Future research can focus on developing techniques to explain model predictions effectively, ensuring trust and usability in practical applications.
2. **Integration with IoT and Sensor Data**: Incorporating real-time sensor data into ML models can further enhance prediction accuracy and enable adaptive control strategies that respond dynamically to changing motor conditions.
3. **Advanced Data Collection and Preprocessing**: Improving methods for collecting and preprocessing diverse datasets, including data from different motor types and operational environments, can enhance model robustness and generalization.
4. **Multi-Physics Modeling**: Integrating ML with multi-physics simulations can create comprehensive models that consider electromagnetic, thermal, and mechanical interactions, providing a holistic understanding of motor behavior.
5. **Lifecycle Optimization**: Expanding ML applications beyond prediction to optimize the entire motor lifecycle, including design iteration, manufacturing processes, and end-of-life considerations, can maximize efficiency and sustainability.
6. **Industry Adoption and Validation**: Further validation of ML models in real-world industrial settings across diverse applications and motor types will be essential to demonstrate their reliability and scalability.

REFERENCES

- **Research Papers and Articles**:

  - K. Liu, J. Wang, and Z. Li, "Modeling and prediction of permanent magnet flux-weakening synchronous motors using machine learning," *IEEE Transactions on Energy Conversion*, vol. 36, no. 1, pp. 257-266, March 2021.
  - L. Li, Y. Wang, J. Wei, and Y. Xing, "Prediction and optimization of permanent magnet synchronous motor torque using machine learning," *Journal of Applied Sciences*, vol. 40, no. 1, pp. 98-105, April 2020.
  - S. N. Jalali, A. A. A. Sadi, and S. J. Vahid, "Permanent magnet synchronous motor parameter estimation using particle swarm optimization and neural networks," *Electric Power Systems Research*, vol. 180, pp. 106123, June 2020.
  - Y. Xu, D. Li, Y. Guan, and L. Wu, "Electric vehicle motor parameter prediction based on an improved support vector machine," *IEEE Access*, vol. 7, pp. 104998-105005, July 2019.

- **Books**:

  - "Electric Motors and Drives: Fundamentals, Types and Applications" by Austin Hughes and Bill Drury (ISBN: 9780081009708).
  - "Machine Learning for Predicting Electric Motor Faults" by Eduardo Gilabert and Manuel Pineda-Sánchez (ISBN: 9783030637943).

- **Conference Proceedings**:

  - Proceedings of the IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS-ITEC), which often includes papers on advanced motor modeling and prediction using machine learning techniques.

- **Industry Reports and Whitepapers**:

  - Reports from industry conferences and workshops focused on electric motor technology, such as those organized by IEEE, SAE International, and various automotive and aerospace industry forums.

- **Online Resources and Journals**:

  - IEEE Xplore Digital Library (https://ieeexplore.ieee.org/)
  - ScienceDirect (https://www.sciencedirect.com/)
  - ResearchGate (https://www.researchgate.net/)

  1.